

Recognizing recurrent neural networks (rRNN): Bayesian inference for recurrent neural networks

Sebastian Bitzer Stefan J. Kiebel

MPI for Human Cognitive and Brain Sciences
Stephanstr. 1a, 04107 Leipzig
Tel.: +49-341-99402426, Fax: +49-341-99402221
bitzer@cbs.mpg.de, kiebel@cbs.mpg.de

June 28, 2011

Abstract

Recurrent neural networks (RNNs) are widely used in computational neuroscience and machine learning applications. In an RNN, each neuron computes its output as a nonlinear function of its integrated input. While the importance of RNNs, especially as models of brain processing, is undisputed, it is also widely acknowledged that the computations in standard RNN models may be an oversimplification of what real neuronal networks compute. Here, we suggest that the RNN approach may be made both neurobiologically more plausible and computationally more powerful by its fusion with Bayesian inference techniques for nonlinear dynamical systems. In this scheme, we use an RNN as a generative model of dynamic input caused by the environment, e.g. of speech or kinematics. Given this generative RNN model, we derive Bayesian update equations that can decode its output. Critically, these updates define a 'recognizing RNN' (rRNN), in which neurons compute and exchange prediction and prediction error messages. The rRNN has several desirable features that a conventional RNN does not have, for example, fast decoding of dynamic stimuli and robustness to initial conditions and noise. Furthermore, it implements a predictive coding scheme for dynamic inputs. We suggest that the Bayesian inversion of recurrent neural networks may be useful both as a model of brain function and as a machine learning tool. We illustrate the use of the rRNN by an ap-

plication to the online decoding (i.e. recognition) of human kinematics.

1 Introduction

Recurrent neural networks (RNN) have been used for many years now to augment nonlinear mappings with a dynamic representation (Pearlmutter, 1989; Williams and Zipser, 1989; Narendran and Parthasarathy, 1990; Jaeger, 2001; Maass et al., 2002), for example, for the classification of sensory input in machine learning. In computational neuroscience, RNNs are extensively used to investigate the dynamic properties of cortical networks (e.g. Buonomano and Maass, 2009; Legenstein and Maass, 2007), to model the measured activity of networks of neurons (e.g. Friston et al., 2003; Kiebel et al., 2006, 2009a; Sotero et al., 2007; Rodrigues et al., 2010) and more generally to model brain processes like perception, memory and attention (e.g. Elman, 1990; Miller and Cohen, 2001; Hamker, 2005). The recurrent connections of these networks capture two of the most prominent features of neuronal networks observed in the brain: Firstly, connections between two neurons are rarely unidirectional but more often bi-directional, potentially via more than one synapse. Secondly, neurons perform highly nonlinear operations, i.e. they transform their input to spiking output. Recurrent neural networks capture both these features where often the input (post-synaptic potentials) and out-

put (action potentials) are replaced by summary measures, i.e. the post-membrane potential function and firing rate. In such a continuous-time RNN, each neuron (often called unit) performs a simple operation: In each moment in time, it applies a nonlinear function to the sum of its input and passes this on to other neurons. This simple mechanism can provide for extremely rich patterns of activity in each neuron, even with a network of small size. Literally thousands of contributions in computational neuroscience and machine learning are based on networks of these firing rate-coding units (Rabinovich et al., 2006; Cessac and Samuelides, 2007).

As powerful as RNNs are as a model class, recent contributions have questioned the usefulness of RNNs to explain neurobiological phenomena observed in real networks of neurons (Spruston, 2008; Mel, 2008; Debanne et al., 2011). Here, we suggest that a simple re-interpretation of the functional role of RNN dynamics leads to a novel and potentially more plausible account of what RNN units may compute: We suggest that neuronal networks serve as Bayesian decoders of dynamics caused by the environment. For example, in action observation, humans decode the kinematics of other people from visual input dynamics. For Bayesian recognition, one needs a so-called generative model for the hidden dynamics of the environment which cause sensory input to the brain. We suggest that RNNs are an ideal generative model for these hidden dynamics in our environment. The task of the recognition system is to decode the sensory input generated by the hidden RNN dynamics. To do this, we derive Bayesian update equations from the generative RNN model and call these 'recognizing RNN'. The difference to a standard RNN is that each unit in the recognizing RNN computes more sophisticated updates involving predictions and prediction error messages from other units in the network. Here we show that a recognizing RNN can decode real-world dynamics (human kinematics) and display several features which can also be observed with real neuronal systems, e.g., the online decoding of hidden dynamics in the environment, computation of predictions and prediction error, robustness to noisy input and fast adaptation to sudden changes in the environment. These features are not only general hallmarks of brain function but may, in principle, also be useful for machine learning applications for

decoding dynamics in an online fashion.

In computational neuroscience, models of recurrently connected networks of neurons, which optimally estimate dynamically changing states from noisy observations, have recently been proposed (Rao, 2004; Denève et al., 2007; Natarajan et al., 2008; Wilson and Finkel, 2009; Boerlin and Denève, 2011). While these models provide important insights, results were reported for relatively restrictive conditions such as linear dynamics (Denève et al., 2007; Wilson and Finkel, 2009; Boerlin and Denève, 2011), discrete states (Rao, 2004; Denève et al., 2007; Boerlin and Denève, 2011), or a one-dimensional state (Natarajan et al., 2008; Wilson and Finkel, 2009; Boerlin and Denève, 2011). Although Natarajan et al. (2008) allow for nonlinear dynamics they assume knowledge of an ideal observer which provides an instantaneous error signal for learning of network connections. Similarly, reservoir computing approaches (Jaeger, 2001; Maass et al., 2002; Verstraeten et al., 2007) rely on a teaching signal which provides a desired output at every point in time during learning. In contrast, we propose an approach combining multi-dimensional, continuous-time hidden nonlinear dynamics where learning proceeds without an externally provided error signal. Our main contribution is to demonstrate that a recognizing RNN is well suited to recognize dynamic stimuli and may be used as a functional model for neuronal ensemble dynamics. In particular, we will illustrate this by showing that the prediction errors computed by a recognizing RNN provide sufficient information to discriminate dynamic stimuli, in an online fashion.

The present approach may also lead to a better understanding of the role of recurrently connected networks of neurons in the brain: predictive coding has been suggested as a theory for hierarchical processing in the brain in which different levels exchange prediction and prediction error messages (Mumford, 1996; Rao and Ballard, 1997, 1999; Friston and Kiebel, 2009). Rao and Ballard (1997) already described RNN-like dynamic models to implement predictive coding for static stimuli. The present approach can be seen as an extension to Rao and Ballard's original work to provide inference for *dynamic* stimuli by resorting to approximate inference methods for nonlinear, continuous dynamic models (Friston et al., 2008).

The remainder of the paper is organised as fol-

lows. In Materials and Methods we (i) present RNNs as generative models, (ii) describe the Bayesian inference framework and (iii) show that dynamic updates of the posterior state critically depend on prediction error. We illustrate the rRNN approach using human motion capture data. In Results we demonstrate that the rRNN can successfully recognize human kinematics and discriminate between different walking styles based on the prediction error of rRNN units.

2 Materials and Methods

In the following, we will describe the two key elements of the present approach: a recurrent neural network (RNN) as a generative model of the sensory dynamics and the Bayesian inference framework to derive the update equations for a recognizing RNN (rRNN). Subsequently, we will apply the rRNN technique to the recognition of human kinematics, for which we describe the kinematic data and the rRNN settings.

To motivate the present approach, we will start with a brief summary of the conventional RNN technique as used in machine learning for classification of stimuli. Note, however, that it is not our aim to compare discrimination performance of conventional and recognizing RNNs. Rather, description of the conventional RNN is given as a reference for understanding the conceptual differences between the two approaches.

2.1 Conventional Recurrent Neural Network

The RNN technique has been used in many machine learning applications such as classification of static or dynamic stimuli, or time-series prediction. This approach has a long history which took off with the development of a supervised learning routine (Pearlmutter, 1989; Williams and Zipser, 1989). Recently, this learning approach has been complemented by the so-called reservoir computing technique (Jaeger, 2001; Maass et al., 2002).

In general, in a conventional RNN, sensory units provide input, which drives the dynamics of the hidden units (see Fig. 1A). Output units readout the result of the dynamic computations based on a mixing of the sensory and hidden states.

An example of such a network is discussed in (Jaeger et al., 2007) where the continuous-time dynamics based on leaky-integrator units is given by

$$\begin{aligned}\dot{x}_i &= f(\mathbf{x})_i \\ &= k_i (-ax_i + \tanh([\mathbf{W}^{\text{in}}\mathbf{y} + \mathbf{W}\mathbf{x} + \mathbf{W}^{\text{fb}}\mathbf{o}]_i))\end{aligned}\quad (1)$$

where x_i is the state of hidden unit $i \in \{1, \dots, H\}$, $\mathbf{y} \in \mathbb{R}^{I \times 1}$ are the states of the input (sensory) units, $\mathbf{o} \in \mathbb{R}^{D \times 1}$ are the states of the readout units, $\mathbf{W} \in \mathbb{R}^{H \times H}$ is a weight matrix defining the interaction between the H hidden units, similarly \mathbf{W}^{in} and \mathbf{W}^{fb} define the connections from the input to the hidden units and the (optional) feedback connections from the readout units, respectively. k_i is a rate constant for unit i and a is the amount of leakage. The output states are determined by

$$\mathbf{o} = \mathbf{V}[\mathbf{x}^T, \mathbf{y}^T]^T \quad (2)$$

where $\mathbf{V} \in \mathbb{R}^{D \times (H+I)}$ is a weight matrix.

In a conventional RNN, the overall flow of information is from sensory to output units, because the RNN serves as a model for neuronal dynamics (hidden states) which are used to compute, e.g., a classification of the sensory input. We now use the same dynamics where we reverse the flow of information to model the generation of sensory dynamics by hidden states of the environment (e.g. body movements cause visual output dynamics).

2.2 Generative Recurrent Neural Network

Our overall aim is to construct a recognition system which can recognize its sensory observations based on its internal dynamics. For a Bayesian recognition system we require a dynamic generative model, for which we choose a RNN. This 'generative recurrent neural network' (gRNN) runs independently of any input and generates sensory data, i.e. observations. Note that, in comparison to a conventional RNN (Eq. 1), here the sensory units become the output of the network while no input units are defined (hence the missing units which acted as output in the conventional RNN). Consequently, the flow of information is reversed in the gRNN and its autonomously running hidden dynamics drive its sensory units (see Fig. 1B). In particular, we

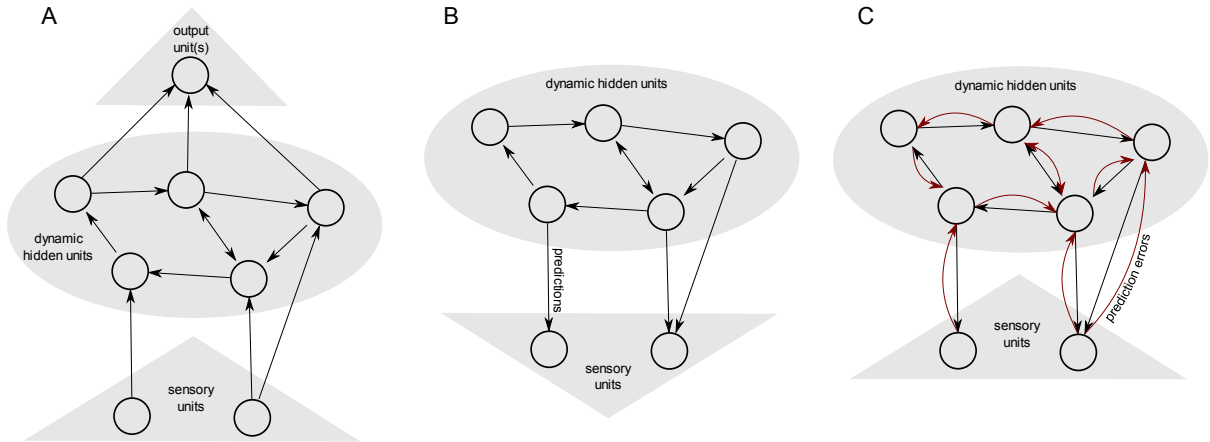


Figure 1: Comparison of different RNN architectures. (A) conventional RNN, (B) generative RNN (gRNN) and (C) recognizing RNN (rRNN). Each RNN has dynamic hidden units, but the overall direction of information flow differs (indicated by the grey triangles). The conventional RNN is designed to compute an output given sensory input. In contrast, the gRNN computes sensory states. Finally, the rRNN computes predictions (black arrows) and prediction error messages (red arrows) to recognize the hidden causes that generated the sensory input.

define a gRNN as

$$\dot{x}_i = f(\mathbf{x})_i = k_i(-ax_i + \tanh([\mathbf{W}\mathbf{x}]_i)) \quad (3)$$

$$\mathbf{y} = \mathbf{V}\mathbf{x} \quad (4)$$

where now $\mathbf{V} \in \mathbb{R}^{D \times H}$ linearly translates hidden states \mathbf{x} into sensory states \mathbf{y} . This gRNN computes sensory output \mathbf{y} as caused by a hidden, dynamic process defined by the RNN dynamics $f(\mathbf{x})$. In the following section, we describe how a recognizing recurrent neural network (rRNN) is constructed from the gRNN using Bayesian inversion. This rRNN receives sensory observations (as in a conventional RNN, Eq. 1) and infers about the hidden states that caused these observations. Effectively, conventional RNN computations are aimed at doing the same (c.f. Fig. 1A,C); however, the update equations of an rRNN are explicitly derived for this recognition task.

2.3 Recognizing Recurrent Neural Network

Generative models like a gRNN (eqs. 3,4) can be used to derive a Bayesian inference system that recognizes input caused by the generative model. In the probabilistic setting, when observations and

state transitions are noisy, or uncertain, Bayesian inversion of the generative model leads to updates of the hidden states which make them an optimal representation of the observations given the model. For example, the well-known Kalman-Bucy filter (Jazwinski, 1970) implements such a Bayesian inversion scheme for linear dynamic processes. The gRNN uses highly nonlinear dynamics (Eq. 3) and, therefore, we require approximate inversion schemes (Jazwinski, 1970; Wan and van der Merwe, 2001; Doucet et al., 2001; Friston et al., 2008; Daunizeau et al., 2009; Friston et al., 2010). Here, we derived the update equations using the D-step of Friston’s dynamic expectation maximization (DEM) framework (Friston et al., 2008). This choice was based on our previous experience with inversion of continuous-time dynamic models using DEM (Kiebel et al., 2009b) but, in principle, other inversion schemes could be used as well. DEM uses generalised coordinates, local linearisation and point-estimates at strategically important positions. See the appendix for a high-level derivation of the algorithm and an explanation of generalised coordinates which are a dynamically extended representation of state variables, the use of which we indicate by a tilde in the subsequent for-

mulas.

In the following, we will briefly describe the key computations performed by DEM. This description is aimed at giving an intuitive description of the update equations governing the rRNN and will allow interpretation of these updates in terms of prediction and prediction error messages.

The most important equation resulting from inversion with DEM describes the evolution of the posterior mode of the hidden states in generalised coordinates and is given by

$$\dot{\tilde{\mathbf{x}}} = \kappa \frac{\partial V(\tilde{\mathbf{x}})}{\partial \tilde{\mathbf{x}}} + \mathbf{D}\tilde{\mathbf{x}}. \quad (5)$$

The motion defined in this equation consists of two parts: 1) $\mathbf{D}\tilde{\mathbf{x}}$ which, in absence of other contributions, implements that the motion of the posterior mode follows its local trajectory represented in generalised coordinates using a derivative operator \mathbf{D} and 2) the derivative of the variational energy $V(\tilde{\mathbf{x}})$ with respect to hidden states which acts as a corrective force to make the motion consistent with the gRNN and the observations. With fixed parameters, the variational energy is the log-joint probability of observations (sensory states) $\tilde{\mathbf{y}}$ and hidden states $\tilde{\mathbf{x}}$ which defines the probabilistic gRNN. In particular, the variational energy is given by

$$\begin{aligned} V(\tilde{\mathbf{x}}) &= \log p(\tilde{\mathbf{y}}, \tilde{\mathbf{x}} | \theta) \\ &= \log p(\tilde{\mathbf{y}} | \tilde{\mathbf{x}}, \theta) + \log p(\mathbf{D}\tilde{\mathbf{x}} | \tilde{\mathbf{f}}, \theta) \\ &= \log p(\tilde{\mathbf{x}} | \tilde{\mathbf{y}}, \theta) + c \end{aligned} \quad (6)$$

where c is a constant, θ is a vector consisting of all parameters of the model and $\tilde{\mathbf{f}}$ are the dynamic predictions defined by Eq. 3 in generalised coordinates. The last term illustrates that the updates are a dynamic form of maximum a posteriori estimation of hidden states. Gaussian distributions are assumed for the state transition and observation densities:

$$p(\mathbf{D}\tilde{\mathbf{x}} | \tilde{\mathbf{f}}, \theta) \sim \mathcal{N}(\tilde{\mathbf{f}}, \tilde{\Sigma}_x) \quad (7)$$

$$p(\tilde{\mathbf{y}} | \tilde{\mathbf{x}}, \theta) \sim \mathcal{N}((\mathbf{I} \otimes \mathbf{V})\tilde{\mathbf{x}}, \tilde{\Sigma}_y) \quad (8)$$

where $(\mathbf{I} \otimes \mathbf{V})\tilde{\mathbf{x}}$ is the predicted sensory state given the hidden states as defined by Eq. 4 in generalised coordinates¹, and $\tilde{\Sigma}_x$ and $\tilde{\Sigma}_y$ are the prior covariances of sensory and hidden states in generalised

¹ \mathbf{I} is the identity matrix with size equal to the number of used generalised coordinates and \otimes is the Kronecker product

coordinates, respectively. This leads to a simple interpretation of the posterior mode updates in terms of prediction errors. In particular, the gradients of these densities with respect to hidden states become

$$\begin{aligned} \frac{\partial \log p(\mathbf{D}\tilde{\mathbf{x}} | \tilde{\mathbf{f}}, \theta)}{\partial \tilde{\mathbf{x}}} &= -\frac{1}{2} \frac{\partial}{\partial \tilde{\mathbf{x}}} \tilde{\varepsilon}_x^T \tilde{\Sigma}_x^{-1} \tilde{\varepsilon}_x \\ &= -\left[\frac{\partial \tilde{\varepsilon}_x}{\partial \tilde{\mathbf{x}}} \right]^T \tilde{\Sigma}_x^{-1} \tilde{\varepsilon}_x \end{aligned} \quad (9)$$

$$\begin{aligned} \frac{\partial \log p(\tilde{\mathbf{y}} | \tilde{\mathbf{x}}, \theta)}{\partial \tilde{\mathbf{x}}} &= -\frac{1}{2} \frac{\partial}{\partial \tilde{\mathbf{x}}} \tilde{\varepsilon}_y^T \tilde{\Sigma}_y^{-1} \tilde{\varepsilon}_y \\ &= -\left[\frac{\partial \tilde{\varepsilon}_y}{\partial \tilde{\mathbf{x}}} \right]^T \tilde{\Sigma}_y^{-1} \tilde{\varepsilon}_y \end{aligned} \quad (10)$$

where the prediction errors are defined as

$$\begin{aligned} \tilde{\varepsilon}_x &= \mathbf{D}\tilde{\mathbf{x}} - \tilde{\mathbf{f}} \\ \tilde{\varepsilon}_y &= \tilde{\mathbf{y}} - (\mathbf{I} \otimes \mathbf{V})\tilde{\mathbf{x}}. \end{aligned} \quad (11)$$

This means that the updates of the posterior hidden states follow the gradient of the prediction error with step sizes determined by the prediction error itself weighted by the prior precisions. The contribution from the prediction error on the sensory states, $\tilde{\varepsilon}_y$, ensures that the sensory states are well explained by the hidden states while the contribution from the prediction error on the hidden states, $\tilde{\varepsilon}_x$, ensures that the posterior dynamics of hidden states as encoded by the generalised coordinates is consistent with the learnt model dynamics. In particular, for the first generalised coordinate, the prediction error

$$\varepsilon_x = \dot{\mathbf{x}} - f(\mathbf{x}) \quad (12)$$

ensures that the posterior velocity corresponds to the learnt, noise free hidden unit dynamics as defined in Eq. 3. Conversely, we will argue below that a consistently large prediction error ε_x provides evidence for an inconsistency between observed and learnt dynamics and can be used to discriminate among different dynamic stimuli.

The question remains how the system got to know a suitable gRNN which generates specific sensory dynamics. In our experiments we let the system learn its generative model by adapting connectivity parameters \mathbf{W} , \mathbf{V} and rate constants \mathbf{k} using an approach which was developed for the identification of dynamical (neural-mass) systems (Friston

et al., 2003; Kiebel et al., 2009a) and is based on maximum a-posteriori estimation of the parameters (Friston, 2002; Friston et al., 2002). See the appendix for details. Note that this initial learning step is not our main point in this paper; any learning approach that successfully learns hidden gRNN dynamics to represent a given dynamic stimulus could be used here (e.g. Wan and Nelson, 2001; Roweis and Ghahramani, 2001; Valpola and Karhunen, 2002; Doucet and Tadić, 2003; Archambeau et al., 2008; Friston et al., 2008; Daunizeau et al., 2009; Kantas et al., 2009; Lazar et al., 2009; Schön et al., 2011), but also standard RNN learning may be used, if the hidden state dynamics is assumed to be deterministic during learning.

2.3.1 Message passing in the rRNN

The updates defined by equations 5, 9, 10 and 11 can be interpreted as network dynamics based on messages sent by sensory and hidden units. Algebraically, this can be seen by exemplarily inspecting the observation density update equation, Eq. 10, for the first generalised coordinate of a single hidden unit i :

$$\begin{aligned} \frac{\partial \log p(\tilde{\mathbf{y}}|\tilde{\mathbf{x}}, \theta)}{\partial x_i} &= -\frac{\partial \tilde{\varepsilon}_y^T}{\partial x_i} \tilde{\Sigma}_y^{-1} \tilde{\varepsilon}_y \\ &= -\sum_j \frac{\partial [\tilde{\varepsilon}_y]_j}{\partial x_i} \left[\tilde{\Sigma}_y^{-1} \tilde{\varepsilon}_y \right]_j \end{aligned} \quad (13)$$

where the sum over j runs over sensory units y_j in generalised coordinates. Note that the partial derivative of the prediction error of sensory unit j with respect to the state of hidden unit i describes how a change in the state of unit i effects the prediction error of unit j . Therefore, the state update for hidden unit i is a weighted sum of these prediction error gradients where each element of this sum corresponds to a 'prediction error message' from a single sensory unit j . To compute the prediction error message a sensory unit first has to compute a prediction. This is done using the forward equation (4) of the gRNN which is a weighted sum of the hidden states $\tilde{\mathbf{x}}$ where the weights are determined by the connectivity of the gRNN. In the following we call the elements of this sum 'prediction messages' which are sent from a hidden unit x_i to a sensory unit y_j . In summary, the update equations define a recognizing RNN (rRNN) where a hidden

unit sends prediction messages to connected sensory and hidden units such that these can compute prediction error messages which are returned to the hidden unit to update its state (see also Fig. 1C). The updates resulting from the dynamics density, Eq. 9, follow the same logic, where the hidden unit x_j takes the place of sensory unit y_j . Each hidden unit, therefore, sends and receives two kinds of messages: prediction and prediction error messages.

2.3.2 Induced connectivity of the rRNN

The connectivity matrices W and V of the gRNN (Eq. 3, 4) are not necessarily the same as in the rRNN. Generally, the rRNN will have all connections of the gRNN plus the corresponding reciprocal connections, plus some additional ones. To see this, note that the prediction error messages in the rRNN in Eq. 13 are 0, when hidden unit i is not connected to sensory unit j , i.e., when hidden unit i has no direct influence on the computation of predictions in sensory unit j (then $\frac{\partial [\tilde{\varepsilon}_y]_j}{\partial x_i} = 0$). Only sensory units which receive a connection from a hidden unit i in the gRNN will contribute messages containing the *derivative* of the prediction error. However, in the rRNN, sensory units j , which are not connected in the gRNN to a hidden unit i , may also contribute messages, containing only their prediction error, through the weights computation $w_j = [\tilde{\Sigma}_y^{-1} \tilde{\varepsilon}_y]_j$. In particular, if the j -th row of the sensory precision matrix, $\tilde{\Sigma}_y^{-1}$, has nonzero entries in positions other than j , e.g., k , the weight of sensory unit j in the update equation (Eq. 13) depends on the prediction error of unit k . In this case, sensory unit k contributes to the update of hidden unit i , even though hidden unit i is not connected to sensory unit k in the gRNN. This means, that there is an additional connection from sensory unit k to hidden unit i in the rRNN.

In conclusion, only if the covariance matrix $\tilde{\Sigma}_y$ is diagonal, the connectivity matrix of sensory to hidden units in the rRNN will only contain those connections which are reciprocal to the hidden to sensory unit connections in the gRNN. Conversely, if there are off-diagonal entries in $\tilde{\Sigma}_y$, there will be corresponding additional connections from sensory to hidden units in the rRNN, relative to the gRNN. The same considerations apply to the connectivity between hidden units. In summary, the connectivity of the rRNN directly follows from the gRNN,

only if the units' states in the gRNN are a priori independent. For simplicity, this case is shown in Fig. 1C and used in the following simulations. Note that a diagonal covariance matrix $\tilde{\Sigma}_y$ is a natural assumption for the present data because we assume that the measurement noise is white and any correlation among observations is caused by the underlying dynamics which are modelled by the RNN dynamics.

2.4 Human Movement Data

We use human movements to demonstrate the properties of the rRNN in the experiments below. The kinematics of humans is highly dynamic and nonlinear through complicated interactions between individual joints. Kinematics, therefore, provides a good example of the kind of complex, dynamically changing, real-world stimuli which can be modelled using rRNNs. Here, we used three walks of the same subject, each of which expresses a different walking style (categorized as 'childish', 'depressed' and 'shy'; freely available from the CMU motion capture database, <http://mocap.cs.cmu.edu>, subject 142, motions 1, 5 and 19). We chose this particular subject because a large range of different movements were available among which we chose the selected walks because of their similar time-scales. The advantage of using motion capture data as compared to video is that we can focus on modelling the kinematics of the subject in terms of changing joint angles without the need to model detailed processing of visual information.

For each walk we removed the global translation of the body and computed the 3D positions of the joints and extremities for all time points. This removed potential 'jumps' introduced by the circularity of the joint angles. As a result we obtained a set of 30 points moving in 3D space (see Fig. 2 for an example). Subsequently, we selected four representative seconds of data starting when the left foot touched the ground for each walk and subsampled the data using 30 frames per second resulting in $N = 120$ data points per walk. These data covered roughly two footsteps for each movement. We then found a common, low-dimensional representation for the three walks using principal component analysis (of all three walks combined) which reduced the dimensionality from 90 dimen-

sions to $D = 5$ (maintaining 95.5% of the original variance). Additionally, we scaled the coordinates of each walk such that the maximum absolute value in each dimension was 1 over all walks. In summary, we obtained for each of the three walks a sequential data set containing five trajectories each consisting of 120 time points, see Fig. 3.

2.4.1 Learning of generative RNNs

For each of the three walks, we constructed one gRNN by learning suitable parameters \mathbf{W} , \mathbf{V} and \mathbf{k} (eqs. 3 and 4) so that the dynamics of each generative RNN replicated the movement data. Each RNN had five sensory units, each of which generated one of the scaled principal component coordinates. In initial tests, we found that a network with $H = 12$ hidden units was the smallest network which gave consistently acceptable learning results and we consequently used this network size in our experiments. These tests also showed that good learning results were obtained, if the hidden units were sparsely connected. In particular, we fixed 2/3 of all connections in \mathbf{W} and 1/3 of all connections in \mathbf{V} to 0. Other entries in \mathbf{W} , the rate constants \mathbf{k} and the initial vector of hidden unit states $\mathbf{x}(0)^t$ were chosen randomly before learning while \mathbf{V} was initially chosen to correctly predict the first data point of a walk given $\mathbf{x}(0)^t$. For details of this initialisation and the learning procedure see the appendix. Note that any learning procedure could have been used here. The main point made by this initial learning step is that a dynamic representation for each walk can be found using RNNs with few units.

The sensory state trajectories of the learnt gRNNs are shown in Fig. 3. Each of the three different walks was learnt well: The amount of variance explained for each walk was 99%, 97% and 97% for the childish, depressed and shy walks, respectively.

3 Results

Here we demonstrate the utility of Bayesian inference for RNNs for online recognition of dynamic stimuli. As a proof of principle we apply the approach to the multi-dimensional, nonlinear kinematics of a walking human. We will first show

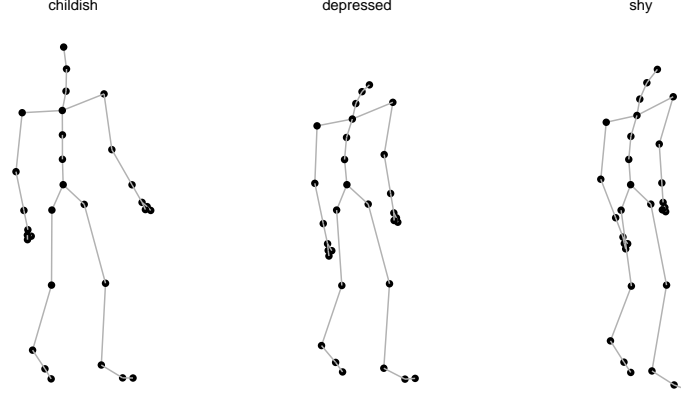


Figure 2: Example frames from the three different walking styles: childish, depressed and shy (left to right). In our experiments we used the first five principal component coordinates of the motion capture 3D joint coordinates (indicated as filled circles) as observation variables. Lines are plotted only for visualisation purposes.

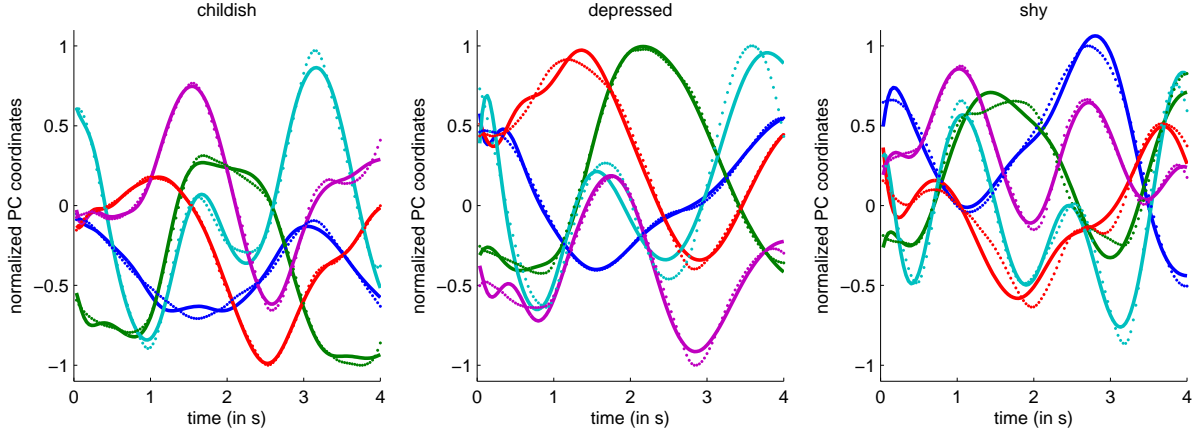


Figure 3: Dynamics of three different human walks and model fits in principle component space (five components). Dotted lines: Original dynamics, Solid lines: Trajectories generated by a gRNN after learning. While the fit between data and its gRNN replications was not perfect, it was sufficiently close such that the gRNN was an appropriate generative model for recognition (see Fig. 5).

that recognizing RNNs quickly and successfully recognize the hidden dynamics, i.e. decode the type of movement. Then, we will demonstrate that the prediction errors of the hidden units can be used to discriminate the three different walks. Finally, we will show that the rRNNs are robust against noisy observations and initial conditions. Note that all of the following experiments with the rRNN use the original motion capture data as observations.

3.1 Fast Recognition of Dynamics

In this section, we show that the rRNN can quickly start recognizing a movement online. In particular, we show that this 'quick response' is robust against the initial hidden states at the beginning of the recognition process. This robustness is obtained despite the fact that gRNNs have a large dependence on their internal initial conditions. This is because RNNs are in general rich dynamic models which are capable of simultaneously representing many different dynamic stimuli depending on their initial conditions (hidden states). We demonstrate this for the gRNN for the childish walk. This gRNN was initialised during learning with the state $\mathbf{x}(0)^l$. When this specific gRNN is started, after learning, in this state, the learnt shy walk is generated as shown in Fig. 3 (left). However, when we initialise the same gRNN with a state $\mathbf{x}(0)^r = \mathbf{x}(0)^l + \epsilon$, which was perturbed by noise of the same size as the natural variability of the hidden states, it generated very different trajectories of sensory states \mathbf{y} as well as hidden states \mathbf{x} as shown in Fig. 4. In other words, for deviating starting conditions, the gRNN generates dynamics that look different from the learnt kinematics and, when plotted in motion capture space, can deviate severely from a natural walk.

In contrast, the rRNN based on this gRNN for the shy walk was robust against such differences in initial conditions. Even though we perturbed the rRNN initial states severely, the rRNN always switched rapidly to the appropriate dynamics which best described the sensory input of a shy walk. In other words, the prediction error updates of the hidden units forced the dynamics on a trajectory which predicted the observed walk. We depict a characteristic example of this quick response behaviour for the rRNN (childish walk) in Fig. 5 (A,B). After only one time step the rRNN accu-

rately predicted all subsequent observations while hidden unit trajectories followed those typical for the learnt gRNN to a large extent. (Note that these results partially depend on an appropriate choice of the prior covariances, see Appendix C.) This means that the rRNN can represent the dynamic repertoire of the gRNN but, in addition, can rapidly switch to the specific dynamic regime that best explains the sensory input.

3.2 Discrimination of Dynamic Stimuli

After learning, we have three different rRNNs, each of which has learnt to predict one of the three walking styles childish, depressed and shy. Here, we will show that the prediction error, on observations $\varepsilon_y = \mathbf{y} - \mathbf{V}\mathbf{x}$ or hidden states $\varepsilon_x = \dot{\mathbf{x}} - f(\mathbf{x})$ (Eq. 12), of all three rRNNs can be used to discriminate between the three different walks. In particular, we will show that the dynamic prediction errors ε_x are smallest for the rRNN that has learnt a specific walking style. This means that a potential readout mechanism can use the relative amplitudes of prediction error of the three rRNNs to decide which of the three walks is currently observed.

Fig. 5 (D-I) shows the response of the rRNN which learnt the childish walk, but now given the depressed and shy walks as observations. Although this rRNN did not learn these walks it represented them well by exploiting alternative dynamics embedded in the twelve unit network. However, the rRNN frequently had to use prediction error on its hidden states to explain away the remaining mismatch between internal predictions and actual input. See Fig. 5 (C,F,I) for this relative increase in prediction error in response to the non-learnt depressed and shy walks. This increase in prediction error when recognizing the two non-learnt walks is consistent over the three different rRNNs and may be used to discriminate dynamic stimuli as shown in Fig. 6 (D-F). For each of the three walks the prediction error was smallest for the rRNN which actually had learnt this specific walk, see also Table 1. The prediction errors on observations showed this effect as well, although not as clearly (Fig. 6 A-C).

We also investigated the effect of learning on the accumulated prediction errors by comparing the prediction errors of the learnt rRNNs with those of

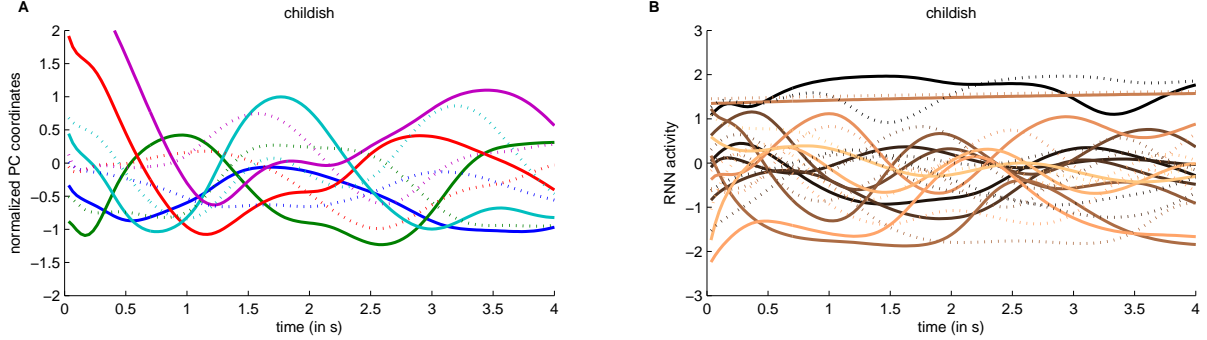


Figure 4: Influence of initial hidden states on dynamics of generative RNN. Shown are trajectories of sensory (A) and hidden (B) states of the generative RNN for two different initial hidden states. Dotted trajectories resulted from initial hidden states used during learning ($\mathbf{x}(0)^l$) while solid trajectories resulted from random initial hidden states ($\mathbf{x}(0)^r$). In this example we used the RNN parameters learnt for walk 1 (childish), but results are qualitatively similar for other RNN parameters.

Table 1: Absolute prediction errors summed over time points and sensory, or hidden states, respectively. Top: sensory state prediction errors. Bottom: dynamic hidden state prediction errors. Each column presents results for each of the three different rRNNs on one of the three data sets childish, depressed and shy. rRNN (random): average accumulated prediction error obtained from 30 random rRNNs (values in parentheses show the minima). rRNN with lowest prediction error on each data set is indicated by bold font. Note that we excluded the first four out of 120 time points from these sums, because the initial transient period otherwise may distort the results.

| sensory state prediction errors | | | |
|---------------------------------|-------------|-------------|-------------|
| | childish | depressed | shy |
| rRNN (childish) | 1.44 | 5.58 | 6.81 |
| rRNN (depressed) | 1.69 | 0.43 | 1.53 |
| rRNN (shy) | 2.89 | 2.66 | 0.56 |
| rRNN (random) | 4.37(2.37) | 4.30(2.20) | 4.21(2.57) |

| hidden state prediction errors | | | |
|--------------------------------|-------------|-------------|-------------|
| | childish | depressed | shy |
| rRNN (childish) | 0.85 | 5.15 | 7.38 |
| rRNN (depressed) | 5.95 | 1.06 | 4.05 |
| rRNN (shy) | 7.39 | 6.44 | 1.48 |
| rRNN (random) | 4.96(3.04) | 4.86(3.13) | 4.65(3.03) |

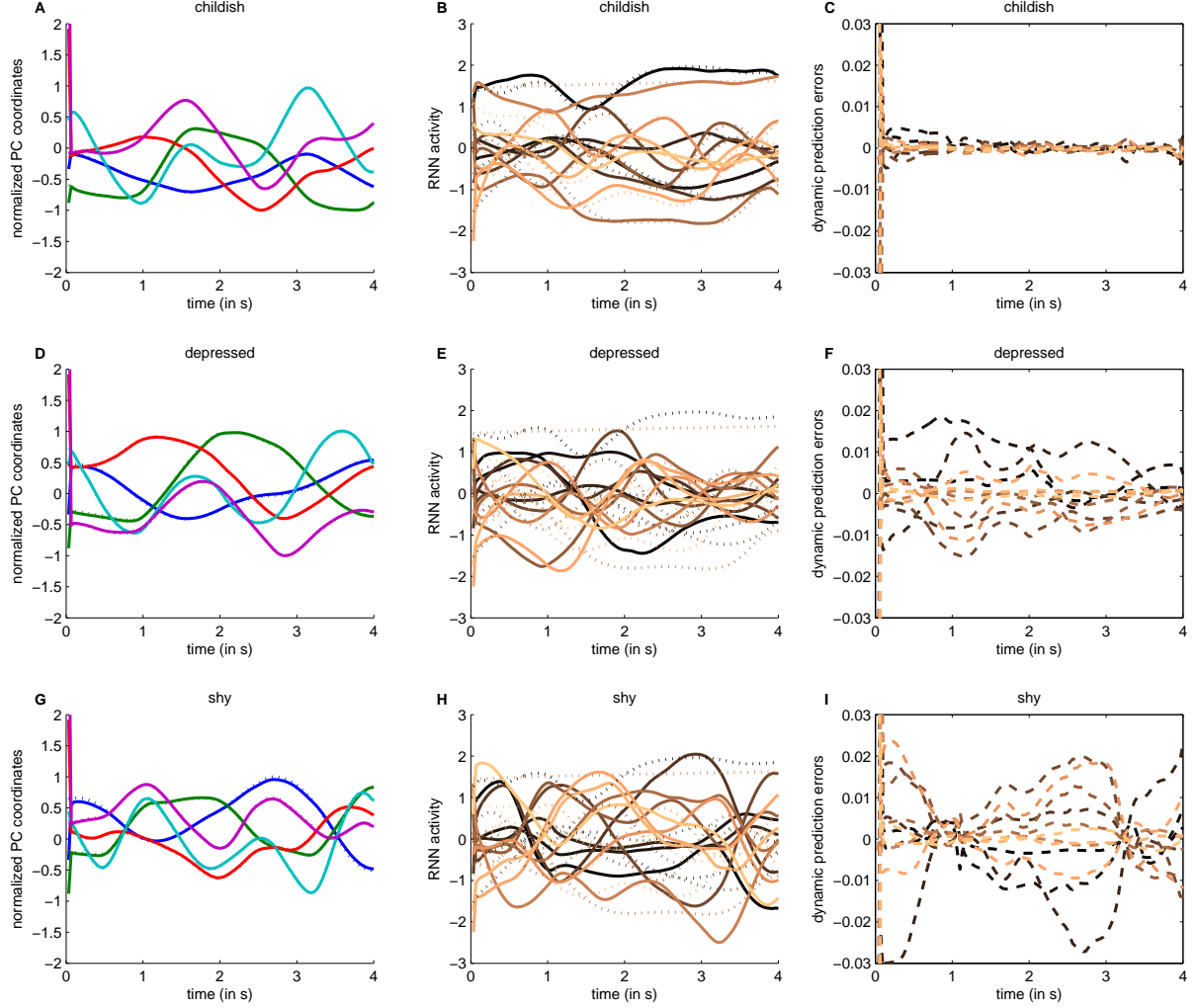


Figure 5: Result of the three different walks recognized by one of the rRNNs (childish walk). (A,D,G) show the presented data (dotted lines) and the predicted sensory states (solid lines). (B,E,H) show the posterior hidden states (solid lines) and, for comparison, the hidden states of the corresponding gRNN when run autonomously from the initial states used during learning (dotted lines, cf. dotted lines in Fig. 4B). (C,F,I) show the dynamic prediction errors of the hidden states (Eq. 12, note that these prediction errors do not correspond to the difference between solid and dotted lines in the middle panels). The different rows of panels correspond to the different walks which were recognized (from top to bottom: childish, depressed and shy). Prediction errors were markedly lower, when the rRNN recognized the walk it was adapted for (C vs. F,I).

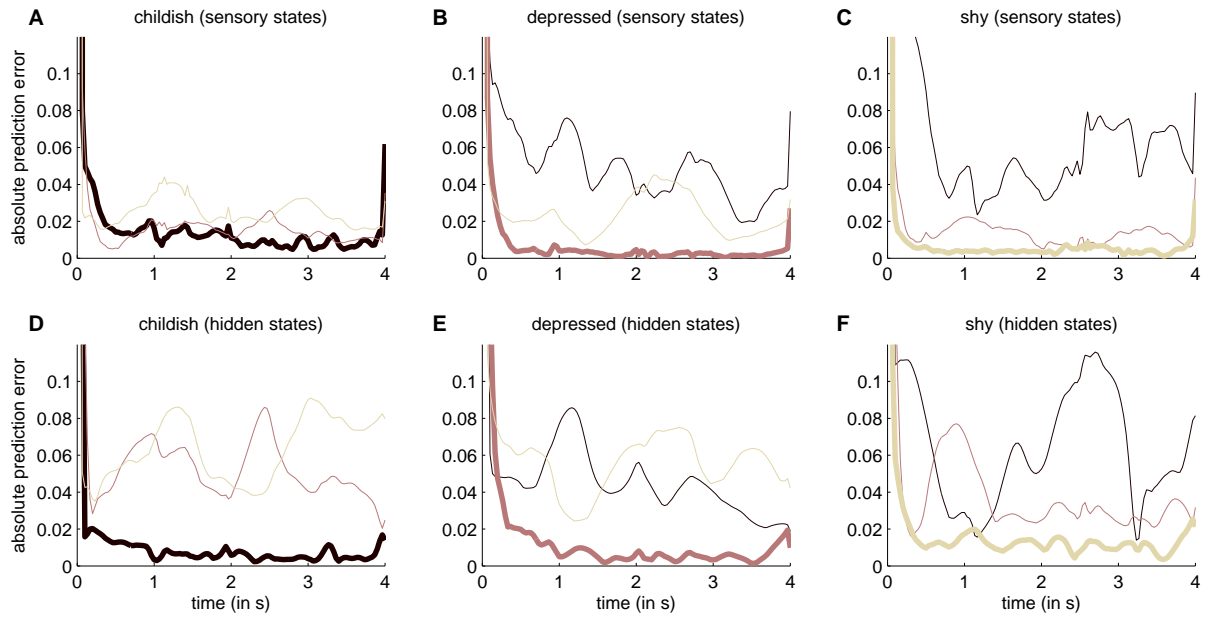


Figure 6: Comparison of absolute prediction errors. Each panel shows summed (over state dimensions) absolute prediction errors of sensory (A-C) and hidden (D-F, Eq. 12) states of the three different rRNNs when data from one of the three different walks were observed. Each rRNN corresponds to one colour (childish: black, depressed: red, shy: yellow). Prediction errors of the rRNN, which has been learnt for the observed type of walk, are indicated by thick lines.

random rRNNs. We generated 30 random rRNNs by drawing random parameters \mathbf{W} , \mathbf{V} and \mathbf{k} while using the same connectivity constraints as for the gRNNs which were used for learning the walks. The accumulated prediction errors for the random rRNNs, thus, give an estimate for the total amount of prediction error expected in a random rRNN, i.e. without learning. As expected, the prediction errors of random rRNNs were always higher than those of the rRNNs with learnt parameters (see Table 1). Furthermore, for non-learnt stimuli, the learnt rRNNs often produced larger prediction errors than random rRNNs. This indicates that learning a specific walk restricts the dynamic repertoire of an rRNN. We conclude that the learning procedure resulted in rRNNs which were suited to discriminate the walks.

In an additional experiment we concatenated data from all three walks into a single sequence to simulate online recognition of three walks, see Fig. 7. The resulting saccade-like, abrupt transitions between walking styles led to a transient increase in prediction errors correctly signalling a discrepancy between predictions and actually observed kinematics. Furthermore, we implemented a simple readout mechanism for dynamic prediction errors using a filter which sums the absolute prediction errors over the last 30 time points and weights recent time points more strongly. This operation smoothes prediction errors temporally and stresses differences that stretch over a similar period as the filter size, see Fig. 7. After each transition, the rRNNs reduced their smoothed prediction errors quickly until the rRNN with parameters learnt for the currently observed walk was the only one for which the magnitude of prediction errors fell below an ad-hoc threshold. This shows that the present approach can be successfully used to recognize a specific walk by choosing the model with the lowest prediction error, after some initial transient has died away.

3.3 Robustness against Noise and Initial Conditions

Here, we demonstrate that the recognition scheme is robust to both noise and variations in initial conditions. We repeated the experiments above for increasing amounts of white observation noise and twelve different, randomly chosen sets of initial con-

ditions, see Fig. 8. We found that the overall magnitude of dynamic prediction errors is proportional to the amount of observation noise. This indicates that observation noise is explained away by prediction errors of both sensory and hidden units. Importantly, the discrimination ability of the three rRNNs is maintained up to moderate amounts of noise, i.e., prediction errors still contained sufficient information to discriminate the three walks. As expected, for large amounts of noise, the contribution from observation noise eventually masked the prediction error contributed by the difference in walks. Also note that the dynamic prediction errors of the learnt rRNNs on their learnt walks (bottom trajectories in the panels of Fig. 8) had very low variability across initial conditions. This means that the rRNN, which was learnt for a specific walk and observes this walk as input, was much less dependent on its initial conditions than the rRNNs learnt on different walks. Yet, the variability of prediction error due to initial conditions within each rRNN was not large enough to influence the result of discrimination of the walks up to moderate amounts of noise. In other words, in our experiments accumulated prediction errors of the rRNN learnt for the current walk were always smaller than those of the other rRNNs (up to moderate amounts of noise), even when beneficial initial conditions for them led to better than average accumulated prediction errors.

4 Discussion

In this paper we have described the ‘recognizing recurrent neural network’ (rRNN), which is a recurrent neural network where each unit computes both predictions and prediction errors to recognize sensory input in a Bayes-optimal fashion. We derived the update equations of both sensory and hidden units by using an approximate Bayesian inference framework for nonlinear dynamical systems, i.e., dynamic expectation maximization (Friston et al., 2008). The rRNN approach is motivated by two considerations: Firstly, in an rRNN, the additional prediction error messages, as compared to a conventional RNN, increase the complexity of the messages passed between units in the network. We propose that these computations may be better descriptions than conventional RNNs of what

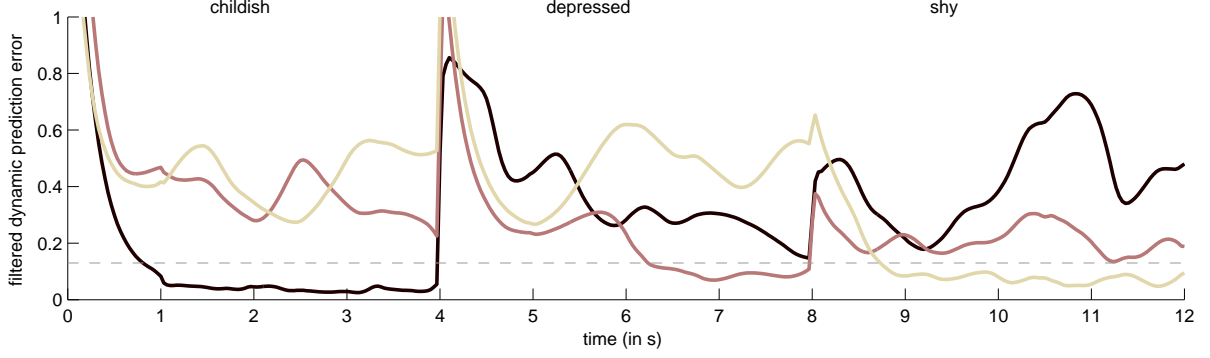


Figure 7: Filtered Dynamic Prediction Errors. We concatenated sensory input from all walks into a single sequence and inferred the hidden states for all three rRNNs. Shown are temporally smoothed, summed absolute prediction errors for the rRNNs learnt on the childish (black), depressed (red) and shy (yellow) walks. We also plotted an ad-hoc decision boundary (dashed, grey line) which could be used to select models with a low prediction error.

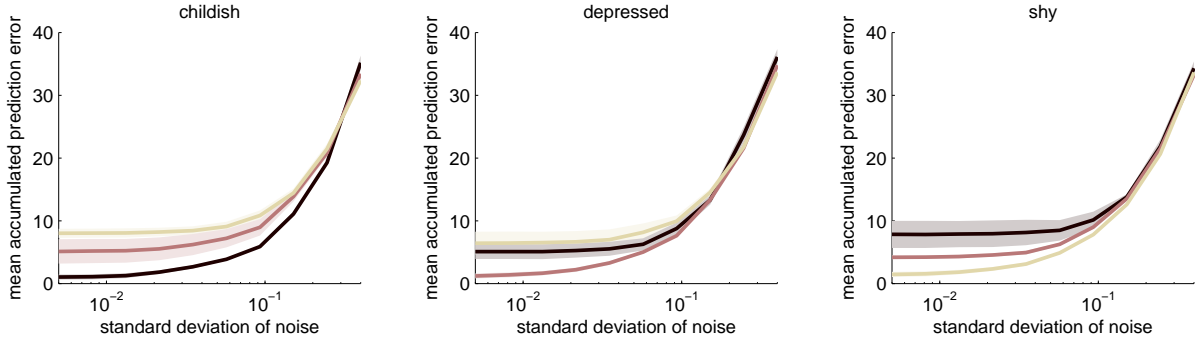


Figure 8: Dependence of dynamic prediction errors on noise and initial conditions. Each panel shows average sums of absolute prediction errors for the three different rRNNs on one of the walks. The averages are over 12 randomly chosen initial states $\mathbf{x}(0)^r$ and shading indicates the region around the mean of twice the standard deviation. The x-axis indicates the standard deviation of independent Gaussian noise added to the principal components of the walks on a log-scale. Note that the exponential increase of prediction errors with noise in the log-plot means that prediction errors depend approximately linear on the observation noise magnitude.

real neuronal ensembles implement. We base this hypothesis on recent findings and theoretical considerations which show that single neurons (and consequently neuronal ensembles) compute much more complex functions than previously thought (Sidiropoulou et al., 2006; Spruston, 2008; Mel, 2008; Pissadaki et al., 2010; Debanne et al., 2011). The general idea is that a single neuron may in principle compute complex, nonlinear and dynamic functions using its spatiotemporal voltage depolarisations and other dynamics like calcium fluctuations (Mel, 2008). Although it is yet unclear how the computation of predictions and prediction errors may map to cellular dynamics, intracellular dynamics may have in principle the computational complexity to perform Bayesian decoding of their synaptic input (Denève, 2008). Secondly, the computation of predictions and prediction errors in an rRNN fits well with the recent reappraisal in cognitive neuroscience that the brain may use a predictive coding approach to perception and cognition (van Wassenhove et al., 2005; Summerfield et al., 2006; Bar, 2009; Friston and Kiebel, 2009). The rRNN approach is a mathematical description of how this predictive coding scheme could be implemented for complex, multi-dimensional dynamic stimuli.

To illustrate that rRNNs may be an interesting model for understanding the brain function of recognition and prediction for naturalistic stimuli, we showed that rRNNs can robustly recognize kinematics as observed with motion capture data. We found that the prediction error computed by an rRNN can be used to recognize and discriminate between different human walking movements in an online fashion. Furthermore, this recognition mechanism is robust against both noise on the observations and variations in the initial state of the rRNN. In other words, rRNNs may be used as functional models for human action observation studies, e.g. (Blake and Shiffrar, 2007). The rRNN approach unifies many important aspects of brain processing such as statistically optimal inference in highly variable and noisy environments, recurrent connections, online recognition of dynamics, and quick adaptation to sudden changes in the environment. Therefore, the rRNN approach may be a useful device to bridge the gap between behaviour-driven models of cognition and neurobiologically plausible models of neuronal ensembles.

The idea to use autonomous RNNs as generative models is not entirely new. In previous work, we have used this approach in system identification where we explained neuroimaging data as generated by a network of cortical nodes, called 'Dynamic Causal Modelling' (DCM) (Friston et al., 2003; Kiebel et al., 2006, 2009a). Critically, the equations governing the dynamics of each node took the form of a rate model as in Eq. 3. The difference to the present approach is that DCM uses specific, highly constrained connectivity schemes based on neural mass models and does not allow for errors in the hidden states. Similarly, we used the present approach (Friston et al., 2008) to model recognition of multi-scale dynamics (Kiebel et al., 2008, 2009b) where the rRNN generalizes these previous contributions by using a more generic generative model (RNNs) and learning of natural stimuli.

To our knowledge, the explicit use of (generic) recurrent neural networks as generative models for recognizing dynamic sensory input using online Bayesian inference has not been described before. Both techniques, Bayesian inference for dynamic stimuli and artificial recurrent neural networks have existed in parallel for many years now (Jazwinski, 1970; Pearlmutter, 1989; Williams and Zipser, 1989; Narendra and Parthasarathy, 1990). We propose the combination of these two approaches in which RNNs act as dynamic models in a nonlinear, Bayesian filtering framework. Indeed, this idea has already been used implicitly in the field of machine learning and control. For example, Connor et al. (1994) used a related approach in the context of autoregressive models to remove outliers from sequences of discrete states which were represented by the hidden states of a RNN. Also, in dual extended Kalman filter methods for RNNs (Wan and Nelson, 2001) an extended Kalman filter is used to estimate RNN hidden states. However, these contributions focus on the usefulness of Bayesian filtering of RNN states to make conventional RNN learning more robust. Here we describe the idea that the combination of RNN equations and filter updates can themselves be interpreted as network equations which are better suited for recognizing dynamic stimuli. Therefore, the present approach also goes beyond previous suggestions of using RNNs as functions approximating the update equations of a nonlinear filter (Parlos et al., 2001), or its output (Ting-Ho Lo, 1994). We, thus, pro-

vide a novel perspective on the role of RNNs also in possible machine learning applications.

We motivated the present approach by considering the potential functional role of recurrently connected neuronal ensembles in cortical processing. This allowed us to address recognition of arbitrary nonlinear dynamics embedded in multidimensional, continuous stimuli - something that has not been reported with spiking neuron models of neuronal coding in recurrent networks (Rao, 2004; Denève et al., 2007; Wilson and Finkel, 2009; Boerlin and Denève, 2011). In contrast, the 'reservoir computing' approach (Jaeger, 2001; Maass et al., 2002; Verstraeten et al., 2007) can recognize the class of stimuli we considered here. The reservoir computing approach has reinvigorated RNN research by establishing that very large networks (hundreds to thousands of units) combined with a simple read-out function can be used to learn and recognize both dynamic and static stimuli. However, reservoir computing approaches typically do not adapt the dynamics within the network and rely on the chance probability that, among the many units, there exist some dynamical regimes which are appropriate generative models of the data. Here, we describe an alternative approach and speculate that small networks of 'smart' rRNN units may be sufficient to recognize dynamic stimuli.

Our use of RNNs as generative models of dynamic stimuli requires learning of their parameters (\mathbf{W} , \mathbf{V} and \mathbf{k} in eqs. 3, 4). In particular, our results depend on learning the connections between hidden units in the recurrent network (\mathbf{W}). This type of learning has been proven to be difficult in the past (Hammer and Steil, 2002). While the learning procedure used here was capable of learning a sufficiently good dynamic representation of the present walks alternative learning approaches may have to be used to achieve similar performance on other data sets. For example, the different principal component coordinates of our walks had similar time scales (Fig. 3). More complex and longer movements may demand the use of hierarchical models and corresponding learning algorithms (Hinton and Salakhutdinov, 2006).

While learning is an important issue with RNNs, we focused on providing a proof-of-principle that the rRNN approach can solve high-level problems such as discriminating visual dynamics in an online fashion. Importantly, our results appear to be ro-

bust against sub-optimally learnt generative RNNs. This can be seen in Fig. 3, which shows a residual difference between learnt trajectories and the input. In other words, we found that prediction error messages were sufficiently informative even though the sensory input observed by the rRNN deviated slightly from the internally predicted dynamics. We also found robustness of the discrimination against white observation noise, see Fig. 8. It is an open question whether the rRNN approach is also robust against structured variations in human movements, e.g. as induced by a variation of a specific movement. We speculate that such variations require a different generative model, either where multiple movements are embedded in a single gRNN or where one uses a hierarchical gRNN similar to the approach used in Taylor and Hinton (2009).

A Bayesian Inversion of Dynamical Models

In this appendix we give a high-level description of the D-step in Friston's dynamic expectation framework (Friston et al., 2008) which leads to Eq. 5 in the main paper.

A.1 Generalised Coordinates

A major component of Friston's approach to stochastic processes is the redefinition of the time-dependent variables in generalised coordinates of motion. For example, one replaces $\mathbf{x}(t)$ with

$$\tilde{\mathbf{x}}(t) = \left[\mathbf{x}(t)^T, \frac{\partial \mathbf{x}(t)^T}{\partial t}, \frac{\partial^2 \mathbf{x}(t)^T}{\partial t^2}, \dots \right]^T \quad (14)$$

and obtains for the probabilistic form of Eq. 3 (dropping the dependence on t for simplicity of writing)

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{x}' = f(\mathbf{x}) + \epsilon^x \\ \frac{\partial^2 \mathbf{x}}{\partial t^2} &= \mathbf{x}'' = \frac{\partial f}{\partial \mathbf{x}} \mathbf{x}' + \epsilon'^x \\ \frac{\partial^3 \mathbf{x}}{\partial t^3} &= \mathbf{x}''' = \frac{\partial f}{\partial \mathbf{x}} \mathbf{x}'' + \epsilon''^x \\ &\vdots \end{aligned} \quad (15)$$

Note that it is assumed that f is locally linear around $\mathbf{x}(t)$ and that differently from the usual

stochastic process models dependencies between noise variables across time are allowed, i.e., it is assumed that the noise at two close points in time correlates and that the noise process $\epsilon^x(t)$ is differentiable sufficiently many times. In generalised coordinates of motion time-dependent variables encode not only a state at the current time, but additionally the future path of states. This is seen when we consider how the continuous representation here can be mapped onto a discrete sequence of N future observations $\mathbf{y} = [y_1, \dots, y_N]^T$ (for simplicity we here show only a single observed variable, i.e., $D = 1$):

$$y_i = \sum_{j=1}^n \frac{\tilde{y}_j(t)}{(j-1)!} (i-t)^{j-1} \quad (16)$$

where n is the highest order of motion considered. We assume that $i = 1, \dots, N$ are the times starting from t at which the data have been sampled. This formula represents a Taylor series approximation making use of the derivatives in $\tilde{\mathbf{y}}$. Friston et al. (2008) have shown that the variance of the noise process quickly becomes very large for high order motions such that only a small number n of generalised coordinates and data points need to be taken into account at any point in time. One also needs to translate discrete data samples into generalised coordinates of motion. This can be done using the inverse operation of Eq. 16. Rewriting Eq. 16 in matrix form gives

$$\mathbf{y} = \mathbf{E}\tilde{\mathbf{y}} \quad e_{ij} = \frac{(i-t)^{j-1}}{(j-1)!} \quad (17)$$

$i \in \{1, \dots, N\}, \quad j \in \{1, \dots, n\}.$

If $N = n$, \mathbf{E} is invertible and one obtains $\tilde{\mathbf{y}}(t) = \mathbf{E}^{-1}\mathbf{y}$. The resulting $\tilde{\mathbf{y}}(t)$ is then used to compute the likelihood of the data at t and make inference over the hidden RNN states $\tilde{\mathbf{x}}(t)$ as described below.

A.2 Dynamic Approximation of the Posterior Mode

In generalised coordinates eqs. 3 and 4 become

$$\tilde{\mathbf{y}} = \tilde{\mathbf{g}} + \tilde{\epsilon}_y \quad \mathbf{D}\tilde{\mathbf{x}} = \tilde{\mathbf{f}} + \tilde{\epsilon}_x \quad (18)$$

where \mathbf{D} is a matrix differentiation operator which shifts coordinates upwards by one element, $\tilde{\mathbf{f}} =$

$[\mathbf{f}^T, \mathbf{f}'^T, \mathbf{f}''^T, \dots]^T$ and $\tilde{\mathbf{g}} = [\mathbf{g}^T, \mathbf{g}'^T, \mathbf{g}''^T, \dots]^T$ are the predicted generalised states and observations, respectively, with $\mathbf{f}' = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{x}'$ and $\mathbf{g}' = \mathbf{V} \mathbf{x}'$ (analogously for higher order terms \mathbf{f}'', \dots). Because $\tilde{\mathbf{g}}$ is linear here, one can write $\tilde{\mathbf{g}} = (\mathbf{I} \otimes \mathbf{V})\tilde{\mathbf{x}}$ where \otimes denotes the Kronecker product and $\mathbf{I} \in \mathbb{R}^{n \times n}$. Based on these equations the log-likelihood of the observations $\tilde{\mathbf{y}}(t)$ is defined as

$$\begin{aligned} \mathcal{L}(t) &= \log p(\tilde{\mathbf{y}}|\theta) \\ &= \log \int p(\tilde{\mathbf{y}}, \tilde{\mathbf{x}}|\theta) d\tilde{\mathbf{x}} \\ &= \log \int p(\tilde{\mathbf{y}}|\tilde{\mathbf{x}}, \theta) p(\tilde{\mathbf{x}}|\tilde{\mathbf{f}}, \theta) d\tilde{\mathbf{x}} \end{aligned} \quad (19)$$

where θ is a placeholder for all parameters in the model. Notice that $p(\tilde{\mathbf{f}}(\tilde{\mathbf{x}})|\mathbf{Y})$, where \mathbf{Y} represents previously seen data, has been approximated as $\delta(\tilde{\mathbf{f}}(\tilde{\mu}))$ - a Dirac delta function at $\tilde{\mathbf{f}}$ evaluated at the previous posterior mode $\tilde{\mu}$ (see below). This means that only the mode is propagated through the dynamics, but not its uncertainty. Friston et al. (2008) then introduce a variational density $q(\tilde{\mathbf{x}})$ (ignoring the density over parameters as learning is not our objective) and make use of Jensen's inequality to obtain

$$\mathcal{L}(t) \geq \int q(\tilde{\mathbf{x}}) \log \frac{p(\tilde{\mathbf{y}}, \tilde{\mathbf{x}}|\theta)}{q(\tilde{\mathbf{x}})} d\tilde{\mathbf{x}} = \mathcal{F}(q, t) \quad (20)$$

where $\mathcal{F}(q, t)$ is the free energy which is a lower bound on the log-likelihood. The aim is to find q such that $\mathcal{L}(t) = \mathcal{F}(q, t)$. In other words, one maximises $\mathcal{F}(q, t)$ with respect to q . This is equivalent to minimising $KL[q(\tilde{\mathbf{x}})||p(\tilde{\mathbf{x}}|\tilde{\mathbf{y}}, \theta)]$, the KL-divergence between variational density and true posterior, i.e., after optimisation q is an approximation of the posterior density over RNN states. In particular, it can be shown (Ghahramani and Beal, 2001; Friston et al., 2008) that the q maximising $\mathcal{F}(q, t)$ is equal to

$$\begin{aligned} q(\tilde{\mathbf{x}}) &= \frac{1}{Z} \exp(V(\tilde{\mathbf{x}})) \\ &= \frac{1}{Z} \exp(\log p(\tilde{\mathbf{y}}, \tilde{\mathbf{x}}|\theta)) \\ &= p(\tilde{\mathbf{x}}|\tilde{\mathbf{y}}, \theta) \end{aligned} \quad (21)$$

where $V(\tilde{\mathbf{x}})$ is called the variational energy. While this equation appears to be a trivial statement, the formulation of q in this way lets us recognize (Friston et al., 2008) that q also is the density defined

by a set of stochastically moving particles at their stationary solution where the movement of a single particle is given by

$$\dot{\mathbf{z}} = \frac{\partial V(\tilde{\mathbf{z}})}{\partial \tilde{\mathbf{z}}} + \Gamma(t) = \frac{\partial \log p(\tilde{\mathbf{y}}, \tilde{\mathbf{z}}|\theta)}{\partial \tilde{\mathbf{z}}} + \Gamma(t) \quad (22)$$

and $\Gamma(t)$ is a random fluctuation. Using this relationship one can find q using Monte Carlo simulation as we can compute the partial derivative of $\log p(\tilde{\mathbf{y}}, \tilde{\mathbf{z}}|\theta)$. However, Friston et al. (2008) simplified this further. In particular, a single particle in generalised coordinates with motion

$$\dot{\mathbf{z}} = \kappa \frac{\partial V(\tilde{\mathbf{z}})}{\partial \tilde{\mathbf{z}}} + \mathbf{D}\tilde{\mathbf{z}} \quad (23)$$

will converge to the mode $\tilde{\mu}$ of V , which is also the mode of the posterior, at a rate proportional to the constant κ (Friston et al., 2008). Given the mode $\tilde{\mu}$ Friston et al. (2008) use a Laplace approximation for the posterior where $q \sim \mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$ is defined to be Gaussian and the covariance $\tilde{\Sigma}$ is found as

$$\tilde{\Sigma}^{-1} = \tilde{\Pi} = - \left. \frac{\partial^2 \log p(\tilde{\mathbf{y}}, \tilde{\mathbf{x}}|\theta)}{\partial \tilde{\mathbf{x}} \partial \tilde{\mathbf{x}}} \right|_{\tilde{\mathbf{x}}=\tilde{\mu}}. \quad (24)$$

This is the inverse of the negative curvature of the posterior evaluated at the mode $\tilde{\mu}$. This completes the derivation of the approximate posterior over RNN states.

Under the approximations made and given the linearity of g one can identify the posterior $p(\tilde{\mathbf{x}}|\tilde{\mathbf{y}}, \theta)$ as being Gaussian exploiting that $p(\tilde{\mathbf{y}}|\tilde{\mathbf{x}}, \theta)$ and $p(\tilde{\mathbf{x}}|\tilde{\mathbf{f}}, \theta)$ are Gaussian. In this case the Laplace approximation is exact. Nevertheless, we retained Friston’s more general form which is also valid for nonlinear g . More importantly, this motivates the dynamic form of estimating the posterior mode in Eq. 23 which allows us to extend the static result above to the dynamic case. In particular, note that all results above were obtained for only a single time point t . However, it can be shown (Friston et al., 2008) that the path integral of the free energy is maximised, if Eq. 21 holds for all t . Naively, this means that one has to integrate the motion of the particle in Eq. 23 until it converges to $\tilde{\mu}(t)$ for each t . However, if the particle converges faster onto $\tilde{\mu}(t)$ than $\tilde{\mu}$ moves itself, a condition which can be ensured by choosing an appropriate rate constant κ , we will be able to track the motion of $\tilde{\mu}$ with a single particle and the dynamics given by

Table 2: Online algorithm for finding the approximate posterior of RNN states.

```

initialise  $\tilde{\mu}(0)$ 
FOR  $t = 1:N$ 
    1) compute predictions  $\tilde{\mathbf{g}}(\tilde{\mu})$  and  $\tilde{\mathbf{f}}(\tilde{\mu})$ 
       from previous  $\tilde{\mu}(t - \Delta t)$ 
    2) find  $\tilde{\mathbf{y}}$  based on  $n$  data points
       closest to  $t$  using Eq. 17
    3) compute gradients of  $V(\tilde{\mathbf{x}})$  using
       predictions,  $\mathbf{D}\tilde{\mu}(t - \Delta t)$  and  $\tilde{\mathbf{y}}$ 
    4) numerically integrate Eq. 23 to
       get new  $\tilde{\mu}(t)$ 
END

```

Eq. 23. Intuitively, the representation in generalised coordinates of motion here helps to converge to a mode which better represents the data as it also takes the local motion (velocity, acceleration, etc.) of the mode into account.

For the purpose of this paper we ignored the approximated covariance and only concentrated on the posterior mode and the corresponding prediction errors. A summary of the resulting algorithm is shown in Table 2. We were able to ignore the covariance, because we assumed network parameters to be fixed during inversion. However, in the full DEM-framework these covariances are needed for the computation of parameter updates.

B Learning of RNN Parameters

We want to adapt the RNN parameters $\mathbf{W}, \mathbf{V}, \mathbf{k}$ such that the observations generated by the RNN defined in eqs. 3,4 fit the data. We mainly follow the approach underlying dynamic causal modelling (Friston et al., 2003; Kiebel et al., 2009a) which is detailed in (Friston, 2002; Friston et al., 2002). This entails an iterative approximation of the parameter posterior based on a first-order Taylor expansion of an observation function $\text{vec}(\mathbf{Y}) = h(\theta)$ which represents the underlying dynamical system. Here, $\mathbf{Y} \in \mathbb{R}^{N \times D}$ contains the observations at all

N time points and $\theta = [\text{vec}(\mathbf{W})^T, \text{vec}(\mathbf{V})^T, \mathbf{k}^T]^T$. The RNN states are enclosed in $h(\theta)$, because the dynamics is assumed to be noise free, i.e., deterministic. Both parameter likelihood and prior are assumed to be Gaussian so that the following gradients of the log-posterior $\mathcal{L} = \log p(\theta|\mathbf{Y}) \propto \log p(\mathbf{Y}|\theta)p(\theta)$ are obtained (cf. Friston, 2002, Eq. 17)

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta} &= \mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{r} + \mathbf{C}_\theta^{-1} (\mu_\theta - \hat{\theta}^{(i)}) \\ \frac{\partial^2 \mathcal{L}}{\partial \theta^2} &\approx \mathbf{J}^T \mathbf{C}_y^{-1} \mathbf{J} + \mathbf{C}_\theta^{-1}.\end{aligned}\quad (25)$$

We use these in a numerical integration scheme for nonlinear dynamical systems to obtain an update of the parameters $d\theta$ based on the model $d\theta/dt = \partial \mathcal{L}/\partial \theta$ and $\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} + d\theta$. Here $\hat{\theta}^{(i)}$ is the maximum a posteriori estimate of the parameters in iteration i , $[\mathbf{J}]_{jk} = \partial[h(\hat{\theta}^{(i)})]_j/\partial \theta_k$ is the Jacobian of h evaluated at $\hat{\theta}^{(i)}$, \mathbf{C}_y is the covariance of the observations and μ_θ and \mathbf{C}_θ are the prior mean and covariance of the parameters, respectively. Finally, $\mathbf{r} = \text{vec}(\mathbf{Y}) - h(\hat{\theta}^{(i)})$ are the residuals of the data not explained by the predictions $h(\hat{\theta}^{(i)})$ which are equivalent to the observation prediction errors ε_y described in the main text. In each iteration one obtains the predictions $h(\hat{\theta}^{(i)})$ by numerical integration of the RNN dynamics and the Jacobian \mathbf{J} using numerical differentiation of $h(\hat{\theta}^{(i)})$.

In our experiments we divided learning into two phases - an initial phase in which we adapted parameters only on local chunks of the data and a final phase in which we used the complete data. We found that the first phase helped to find a better initialisation of $\hat{\theta}$ for the optimisation on the whole data set in the second phase of learning. In the first phase we split the data into seven overlapping, equal size chunks and ran two passes through all chunks where we ran only two iterations of the update procedure described above per chunk and pass. In the second phase we ran 25 iterations with a fast, approximate numerical integration scheme for h and subsequently another 4 iterations with a slower, but more accurate scheme. While our choices for the number of chunks, passes and iterations led to good results, we expect that many other values may be chosen equivalently.

Embedded in each iteration there is also an expectation maximization (EM)-like update of hyperparameter λ which determines the amount of noise

on the observations $\mathbf{C}_y = e^\lambda \mathbf{I}$ during learning. We refer the reader to (Friston, 2002; Friston et al., 2002) for details. λ was initialised as -32. The hyperprior for λ was Gaussian $\lambda \sim \mathcal{N}(-\log(\bar{s}^2), 1/8)$ where \bar{s}^2 is the average variance of the observation variables in the data.

We initialised the parameters contained in $\theta^{(0)}$ as follows. The elements of $\mathbf{k}^{(0)}$ were chosen uniformly in the interval $[1/8, 3/8]$. Randomly chosen 2/3 of all elements in $\mathbf{W}^{(0)}$ were fixed at 0, the remaining were drawn randomly from a standard normal distribution. Furthermore, following (Jaeger et al., 2007), we scaled the resulting matrix by $\mathbf{W} = 1/(0.95\delta)\mathbf{W}^{(0)}$ to bring the initial RNN dynamics to a useful dynamical regime. δ here is the largest absolute eigenvalue of the matrix $[\bar{k}\mathbf{W}^{(0)} - (1 - \bar{k}a)\mathbf{I}]$ where a is the leakage (cf. Eq. 1) and $\bar{k} = 1/4$ is the expected value of k_i for any i . The initial states $x_j(0)^l$ were chosen uniformly in $[-2, 2]$ for all j . We then found $\mathbf{V}^{(0)}$ as the solution to the underdetermined system of equations $\mathbf{y}(0) = \mathbf{V}^{(0)}\mathbf{x}(0)^l$ using Matlab's backslash operator, i.e., we found the least squares solution for $\mathbf{V}^{(0)}$ with most elements of $\mathbf{V}^{(0)}$ equal to zero. A randomly chosen subset of these zero elements were also fixed during learning. The number of fixed elements was 1/3 of the total number of elements.

In the initial learning phase we set the mean of the parameter prior to the described initialisation of the parameters $\mu_\theta = \theta^{(0)}$. In the subsequent learning phase we set μ_θ to the result of the 1st phase. The covariances of the prior parameter distribution were chosen to be diagonal, but also differed in the two phases of learning. In the initial phase we set the variances associated with the elements of \mathbf{W} to $1.6 \cdot 10^5$ while we set the variances for \mathbf{V} and \mathbf{k} to 0.018 and 0.135, respectively. This enforced particularly the adaptation of the dynamical parameters. For learning on the full data set we chose these variances to be 7.389, 1 and 1 for \mathbf{W} , \mathbf{V} and \mathbf{k} , respectively.

C Prior Covariances

For the rRNN the prior covariances, $\tilde{\Sigma}_y$ and $\tilde{\Sigma}_x$, modulate the size of updates of the posterior (cf. eqs. 10 and 9) and influence the result of the Bayesian inversion. Intuitively, for large prior (co-)variances, i.e., a large amount of a priori expected

noise, smaller updates are made and larger prediction errors are tolerated. The amount of noise here has to be seen in comparison to the variance of the unperturbed states of the units in the gRNN. For the sensory states this corresponds to the variance of the movement data. The standard deviation of the sensory states across all walks averaged over the five input dimensions was 0.38 while the standard deviation of the corresponding changes in hidden states averaged over the 12 hidden units was 0.04. In our simulations we assumed isotropic prior noise and correspondingly chose covariances of the form $\Sigma_y = \sigma_y^2 \mathbf{I}$ ² where \mathbf{I} is the identity matrix and σ_y is our choice of standard deviation. We chose $\sigma_y = 0.3$ and $\sigma_x = 0.1$ for sensory and hidden states, respectively. This means that we tolerated only relatively small prediction errors on sensory states while allowing for relatively larger prediction errors on changes of hidden states. This choice implements the natural prior belief that the variability of walk observations is mainly determined by the variability of the underlying dynamics.

References

- C. Archambeau, M. Opper, Y. Shen, D. Cornford, and J. Shawe-Taylor. Variational inference for diffusion processes. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 17–24. MIT Press, Cambridge, MA, 2008.
- M. Bar. The proactive brain: memory for predictions. *Philos Trans R Soc Lond B Biol Sci*, 364(1521):1235–1243, May 2009. doi: 10.1098/rstb.2008.0310. URL <http://dx.doi.org/10.1098/rstb.2008.0310>.
- R. Blake and M. Shiffrar. Perception of human motion. *Annu Rev Psychol*, 58:47–73, 2007. doi: 10.1146/annurev.psych.57.102904.190152. URL <http://dx.doi.org/10.1146/annurev.psych.57.102904.190152>.
- M. Boerlin and S. Denève. Spike-based population coding and working memory. *PLoS Comput Biol*, 7(2):e1001080, 02 2011. doi: 10.1371/journal.pcbi.1001080. URL <http://dx.doi.org/10.1371/journal.pcbi.1001080>. Deneve (just so that I find it when searching for Deneve without the accent).
- D. V. Buonomano and W. Maass. State-dependent computations: spatiotemporal processing in cortical networks. *Nat Rev Neurosci*, 10(2):113–125, Feb 2009. doi: 10.1038/nrn2558. URL <http://dx.doi.org/10.1038/nrn2558>.
- B. Cessac and M. Samuelides. From neuron to neural networks dynamics. *The European Physical Journal - Special Topics*, 142:7–88, 2007. ISSN 1951-6355. URL <http://dx.doi.org/10.1140/epjst/e2007-00058-2>. 10.1140/epjst/e2007-00058-2.
- J. T. Connor, R. D. Martin, and L. E. Atlas. Recurrent neural networks and robust time series prediction. *Neural Networks, IEEE Transactions on*, 5(2):240–254, 1994. doi: 10.1109/72.279188.
- J. Daunizeau, K. Friston, and S. Kiebel. Variational bayesian identification and prediction of stochastic nonlinear dynamic causal models. *Physica D: Nonlinear Phenomena*, 238(21):2089 – 2118, 2009. ISSN 0167-2789. doi: DOI:10.1016/j.physd.2009.08.002. URL <http://www.sciencedirect.com/science/article/B6TVK-4X0878M-1/2/d6d9ac7898c5573bd8a33605a7eaf89a>.
- D. Debanne, E. Campanac, A. Bialowas, E. Carlier, and G. Alcaraz. Axon physiology. *Physiol Rev*, 91(2):555–602, Apr 2011. doi: 10.1152/physrev.00048.2009. URL <http://dx.doi.org/10.1152/physrev.00048.2009>.
- S. Denève. Bayesian spiking neurons i: inference. *Neural Comput*, 20(1):91–117, Jan 2008. doi: 10.1162/neco.2008.20.1.91. URL <http://dx.doi.org/10.1162/neco.2008.20.1.91>.
- S. Denève, J.-R. Duhamel, and A. Pouget. Optimal sensorimotor integration in recurrent cortical networks: a neural implementation of kalman filters. *J Neurosci*, 27(21):5744–5756, May 2007. doi: 10.1523/JNEUROSCI.3985-06.2007. URL <http://dx.doi.org/10.1523/JNEUROSCI.3985-06.2007>.

²To see how these standard covariance matrices are translated into generalised coordinates we refer the reader to (Friston et al., 2008, p. 860).

- A. Doucet and V. Tadić. Parameter estimation in general state-space models using particle methods. *Annals of the Institute of Statistical Mathematics*, 55:409–422, 2003. ISSN 0020-3157. URL <http://dx.doi.org/10.1007/BF02530508>. 10.1007/BF02530508.
- A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.
- J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 1551-6709. doi: 10.1207/s15516709cog1402_1. URL http://dx.doi.org/10.1207/s15516709cog1402_1.
- K. Friston and S. Kiebel. Predictive coding under the free-energy principle. *Philos Trans R Soc Lond B Biol Sci*, 364(1521):1211–1221, May 2009. doi: 10.1098/rstb.2008.0300. URL <http://dx.doi.org/10.1098/rstb.2008.0300>.
- K. Friston, N. Trujillo-Barreto, and J. Daunizeau. DEM: A variational treatment of dynamic systems. *NeuroImage*, 41(3):849 – 885, 2008. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2008.02.054. URL <http://www.sciencedirect.com/science/article/B6WNP-4S19RH8-1/2/48c1aa6f77adaeeaba06cd57573a986d>.
- K. Friston, K. Stephan, B. Li, and J. Daunizeau. Generalised filtering. *Mathematical Problems in Engineering*, 2010:Article ID 621670, 2010. doi: 10.1155/2010/621670.
- K. J. Friston. Bayesian estimation of dynamical systems: an application to fMRI. *Neuroimage*, 16(2):513–530, Jun 2002. doi: 10.1006/nimg.2001.1044. URL <http://dx.doi.org/10.1006/nimg.2001.1044>.
- K. J. Friston, W. Penny, C. Phillips, S. Kiebel, G. Hinton, and J. Ashburner. Classical and bayesian inference in neuroimaging: theory. *Neuroimage*, 16(2):465–483, Jun 2002. doi: 10.1006/nimg.2002.1090. URL <http://dx.doi.org/10.1006/nimg.2002.1090>.
- K. J. Friston, L. Harrison, and W. Penny. Dynamic causal modelling. *Neuroimage*, 19(4):1273–1302, Aug 2003.
- Z. Ghahramani and M. J. Beal. Propagation algorithms for variational bayesian learning. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 507–513. MIT Press, 2001.
- F. H. Hamker. The reentry hypothesis: the putative interaction of the frontal eye field, ventrolateral prefrontal cortex, and areas v4, it for attention and eye movement. *Cereb Cortex*, 15(4):431–447, Apr 2005. doi: 10.1093/cercor/bhh146. URL <http://dx.doi.org/10.1093/cercor/bhh146>.
- B. Hammer and J. J. Steil. Tutorial: Perspectives on learning with rnns. In *Proceedings of European Symposium on Artificial Neural Networks (ESANN)*, pages 357–368. d-side publi., 2002.
- S. Haykin, editor. *Kalman Filtering and Neural Networks*. John Wiley & Sons, Inc., 2001. doi: 10.1002/0471221546.
- G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, Jul 2006. doi: 10.1126/science.1127647. URL <http://dx.doi.org/10.1126/science.1127647>.
- H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology, 2001.
- H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, Apr 2007. doi: 10.1016/j.neunet.2007.04.016. URL <http://dx.doi.org/10.1016/j.neunet.2007.04.016>.
- A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.
- N. Kantas, A. Doucet, S. S. Singh, and J. M. Maciejowski. Overview of sequential monte carlo methods for parameter estimation on general state space models. In *Proc. 15th IFAC Symposium on System Identification (SYSID)*, Saint-Malo, France, 2009. URL <http://publications.eng.cam.ac.uk/16156/>.

- S. J. Kiebel, O. David, and K. J. Friston. Dynamic causal modelling of evoked responses in eeg/meg with lead field parameterization. *Neuroimage*, 30(4):1273–1284, May 2006. doi: 10.1016/j.neuroimage.2005.12.055. URL <http://dx.doi.org/10.1016/j.neuroimage.2005.12.055>.
- S. J. Kiebel, J. Daunizeau, and K. J. Friston. A hierarchy of time-scales and the brain. *PLoS Comput Biol*, 4(11):e1000209, Nov 2008. doi: 10.1371/journal.pcbi.1000209. URL <http://dx.doi.org/10.1371/journal.pcbi.1000209>.
- S. J. Kiebel, M. I. Garrido, R. Moran, C.-C. Chen, and K. J. Friston. Dynamic causal modeling for eeg and meg. *Hum Brain Mapp*, 30(6):1866–1876, Jun 2009a. doi: 10.1002/hbm.20775. URL <http://dx.doi.org/10.1002/hbm.20775>.
- S. J. Kiebel, K. von Kriegstein, J. Daunizeau, and K. J. Friston. Recognizing sequences of sequences. *PLoS Comput Biol*, 5(8):e1000464, Aug 2009b. doi: 10.1371/journal.pcbi.1000464. URL <http://dx.doi.org/10.1371/journal.pcbi.1000464>.
- A. Lazar, G. Pipa, and J. Triesch. Sorn: a self-organizing recurrent neural network. *Front Comput Neurosci*, 3:23, 2009. doi: 10.3389/neuro.10.023.2009. URL <http://dx.doi.org/10.3389/neuro.10.023.2009>.
- R. Legenstein and W. Maass. What makes a dynamical system computationally powerful? In S. Haykin, J. C. Principe, T. J. Sejnowski, and J. G. McWhirter, editors, *New Directions in Statistical Signal Processing: From Systems to Brains*, pages 127–154. MIT Press, 2007.
- W. Maass, T. Natschlger, and H. Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput*, 14(11):2531–2560, Nov 2002. doi: 10.1162/089976602760407955. URL <http://dx.doi.org/10.1162/089976602760407955>.
- B. W. Mel. Why have dendrites? a computational perspective. In G. Stuart, N. Spruston, and M. Häusser, editors, *Dendrites*, chapter 16. Oxford University Press, 2nd edition, 2008.
- E. K. Miller and J. D. Cohen. An integrative theory of prefrontal cortex function. *Annu Rev Neurosci*, 24:167–202, 2001. doi: 10.1146/annurev.neuro.24.1.167. URL <http://dx.doi.org/10.1146/annurev.neuro.24.1.167>.
- D. Mumford. Pattern theory: A unifying perspective. In D. C. Knill and W. Richards, editors, *Perception as Bayesian Inference*. Cambridge University Press, 1996.
- K. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *Neural Networks, IEEE Transactions on*, 1(1):4–27, mar 1990. ISSN 1045-9227. doi: 10.1109/72.80202.
- R. Natarajan, Q. J. M. Huys, P. Dayan, and R. S. Zemel. Encoding and decoding spikes for dynamic stimuli. *Neural Computation*, 20(9):2325–2360, 2008. doi: 10.1162/neco.2008.01-07-436. URL <http://www.mitpressjournals.org/doi/abs/10.1162/neco.2008.01-07-436>.
- A. Parlos, S. Menon, and A. Atiya. An algorithmic approach to adaptive state filtering using recurrent neural networks. *Neural Networks, IEEE Transactions on*, 12(6):1411–1432, nov 2001. ISSN 1045-9227. doi: 10.1109/72.963777.
- B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269, 1989. doi: 10.1162/neco.1989.1.2.263. URL <http://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.2.263>.
- E. K. Pissadaki, K. Sidiropoulou, M. Reczko, and P. Poirazi. Encoding of spatio-temporal input characteristics by a cal pyramidal neuron model. *PLoS Comput Biol*, 6(12):e1001038, 2010. doi: 10.1371/journal.pcbi.1001038. URL <http://dx.doi.org/10.1371/journal.pcbi.1001038>.
- M. I. Rabinovich, P. Varona, A. I. Selverston, and H. D. I. Abarbanel. Dynamical principles in neuroscience. *Rev. Mod. Phys.*, 78(4):1213–1265, Nov 2006. doi: 10.1103/RevModPhys.78.1213.
- R. P. Rao and D. H. Ballard. Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Comput*, 9(4):721–763, May 1997.

- R. P. Rao and D. H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat Neurosci*, 2(1):79–87, Jan 1999. doi: 10.1038/4580. URL <http://dx.doi.org/10.1038/4580>.
- R. P. N. Rao. Bayesian computation in recurrent neural circuits. *Neural Comput*, 16(1):1–38, Jan 2004. doi: 10.1162/08997660460733976.
- S. Rodrigues, A. V. Chizhov, F. Marten, and J. R. Terry. Mappings between a macroscopic neural-mass model and a reduced conductance-based model. *Biol Cybern*, 102(5):361–371, May 2010. doi: 10.1007/s00422-010-0372-z. URL <http://dx.doi.org/10.1007/s00422-010-0372-z>.
- S. Roweis and Z. Ghahramani. Learning nonlinear dynamical systems using the expectation-maximization algorithm. In Haykin (2001). doi: 10.1002/0471221546.
- T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39 – 49, 2011. ISSN 0005-1098. doi: 10.1016/j.automatica.2010.10.013. URL <http://www.sciencedirect.com/science/article/pii/S0005109810004279>.
- K. Sidiropoulou, E. K. Pissadaki, and P. Poirazi. Inside the brain of a neuron. *EMBO Rep*, 7(9):886–892, Sep 2006. doi: 10.1038/sj.embor.7400789. URL <http://dx.doi.org/10.1038/sj.embor.7400789>.
- R. C. Sotero, N. J. Trujillo-Barreto, Y. Iturria-Medina, F. Carbonell, and J. C. Jimenez. Realistically coupled neural mass models can generate eeg rhythms. *Neural Comput*, 19(2):478–512, Feb 2007. doi: 10.1162/neco.2007.19.2.478. URL <http://dx.doi.org/10.1162/neco.2007.19.2.478>.
- N. Spruston. Pyramidal neurons: dendritic structure and synaptic integration. *Nat Rev Neurosci*, 9(3):206–221, Mar 2008. doi: 10.1038/nrn2286. URL <http://dx.doi.org/10.1038/nrn2286>.
- C. Summerfield, T. Egner, M. Greene, E. Koechlin, J. Mangels, and J. Hirsch. Predictive codes for forthcoming perception in the frontal cortex. *Science*, 314(5803):1311–1314, Nov 2006. doi: 10.1126/science.1132028. URL <http://dx.doi.org/10.1126/science.1132028>.
- G. W. Taylor and G. E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.
- J. Ting-Ho Lo. Synthetic approach to optimal filtering. *Neural Networks, IEEE Transactions on*, 5(5):803–811, sep 1994. ISSN 1045-9227. doi: 10.1109/72.317731.
- H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002. doi: 10.1162/089976602760408017. URL <http://www.mitpressjournals.org/doi/abs/10.1162/089976602760408017>.
- V. van Wassenhove, K. W. Grant, and D. Poeppel. Visual speech speeds up the neural processing of auditory speech. *Proc Natl Acad Sci U S A*, 102(4):1181–1186, Jan 2005. doi: 10.1073/pnas.0408949102. URL <http://dx.doi.org/10.1073/pnas.0408949102>.
- D. Verstraeten, B. Schrauwen, M. D’Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Netw*, 20(3):391–403, Apr 2007. doi: 10.1016/j.neunet.2007.04.003. URL <http://dx.doi.org/10.1016/j.neunet.2007.04.003>.
- E. A. Wan and A. T. Nelson. Dual extended kalman filter methods. In Haykin (2001). doi: 10.1002/0471221546.
- E. A. Wan and R. van der Merwe. The unscented kalman filter. In Haykin (2001). doi: 10.1002/0471221546.
- R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270. URL <http://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.2.270>.

R. Wilson and L. Finkel. A neural implementation of the kalman filter. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2062–2070. 2009.